รายงานวิชาปฏิบัติการ

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

รหัสวิชา ____242-301____ ตอน __01__ วัน__**อังคาร**__

รหัสหัวข้อปฏิบัติการ _____2HA08_____

ชื่อหัวข้อปฏิบัติการ____ Simple music box using FPGA _____

วันที่ลงปฏิบัติการ ____7 พฤศจิกายน 2560_____

อาจารย์ผู้สอน _____**อาจารย์สุธาสินีย์ เสน่ห์เสาวรส**_____

กลุ่มที่ ___8____

ผู้จัดทำรายงานชื่อ ___**นายปณิธาน ดวงขวัญ** _____ รหัส __5735512036_____

ผู้ร่วมงาน ชื่อ ___**นางสาวรัตนาพร ไทยเกิด**_____ รหัส __5735512047_____

ชื่อ _____รหัส _____

สำหรับเจ้าหน้าที่

วันที่ตรวจรับ _____

ลงชื่อ _____

การทดลองที่ 3HA08

Simple music box using FPGA

เครื่องมือและอุปกรณ์

1. โปรแกรม Xilinx
2. บอร์ด FPGA รุ่น XC3S200-TQ144C
3. คอมพิวเตอร์
4. ลำโพง

การทดลองที่ 1

```
21  module c1(
22      input clk,
23      input onoff,
24      input [6:0] note,
25      output speaker
26      );
27
28    parameter doo = (25000000/262/2);
29    parameter re = (25000000/294/2);          //config frequncey
30    parameter mi = (25000000/330/2);
31    parameter fa = (25000000/349/2);
32    parameter sol = (25000000/392/2);
33    parameter la = (25000000/440/2);
34    parameter si = (25000000/494/2);
35
36    reg s_do,s_re,s_mi,s_fa,s_sol,s_la,s_si;
37    reg [31:0] cnt,cnt1,cnt2,cnt3,cnt4,cnt5,cnt6,cnt7;    //config variable
38    reg sp;
39    assign speaker = sp;
40
41    reg [5:0] state = 0;
42    always@(state)                    //case -> in play note
43    begin
44       if(onoff == 0) sp <= 0;
45       else
46       case(note)
47          7'b0000001 : sp <= s_do;
48          7'b0000010 : sp <= s_re;
49          7'b0000100 : sp <= s_mi;
50          7'b0001000 : sp <= s_fa;
51          7'b0010000 : sp <= s_sol;
52          7'b0100000 : sp <= s_la;
53          7'b1000000 : sp <= s_si;
54
55
56       endcase
57    end
58
59    always @(posedge clk)                    //config clock in each note
```

```
60    begin
61       if (cnt1 == 0) begin cnt1 <= doo - 1 ; s_do <= ~s_do; end
62          else cnt1 <= cnt1 - 1;   end
63
64
65    always @(posedge clk)
66    begin
67       if (cnt2 == 0) cnt2 <= re - 1; else cnt2 <= cnt2 - 1; end
68    always @(posedge clk)
69       if(cnt2 == 0) s_re <= ~s_re;
70
71    always @(posedge clk)
72    begin
73       if (cnt3 == 0) begin cnt3 <= mi - 1; s_mi <= ~s_mi; end else cnt3 <= cnt3 - 1; end
74
75
76    always @(posedge clk)
77    begin
78       if (cnt4 == 0) begin cnt4 <= fa - 1; s_fa <= ~s_fa; end else cnt4 <= cnt4 - 1; end
79
80
81    always @(posedge clk)
82    begin
83       if (cnt5 == 0) begin cnt5 <= sol - 1;  s_sol <= ~s_sol; end else cnt5 <= cnt5 - 1; end
84
85
86    always @(posedge clk)
87    begin
88       if (cnt6 == 0) begin cnt6 <= la - 1; s_la <= ~s_la; end else cnt6 <= cnt6 - 1; end
89
90
91    always @(posedge clk)
92    begin
93       if (cnt7 == 0) begin cnt7 <= si - 1; s_si <= ~s_si; end else cnt7 <= cnt7 - 1; end
94
95
96
97    always@(posedge clk)                                          //config change state
98    if(onoff == 0||state == 37) state <= 0 ;else
```

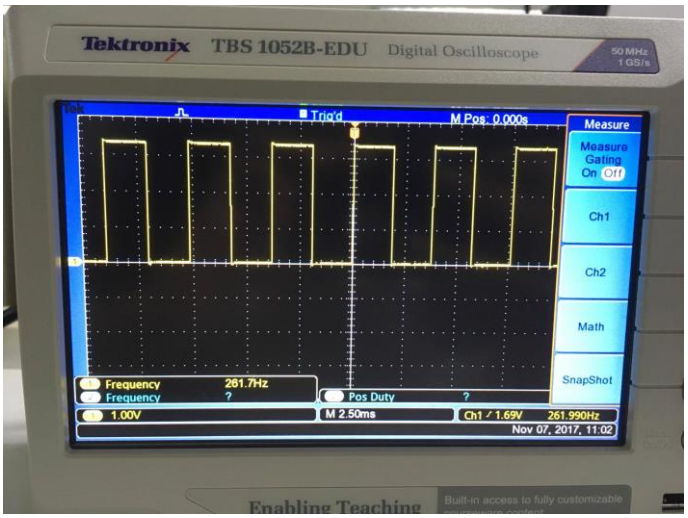
```
99    begin
100      if(cnt == 10000000 ) begin state <= state + 1 ; cnt <=0 ; end
101      else cnt <= cnt + 1; end
102
103   endmodule
```

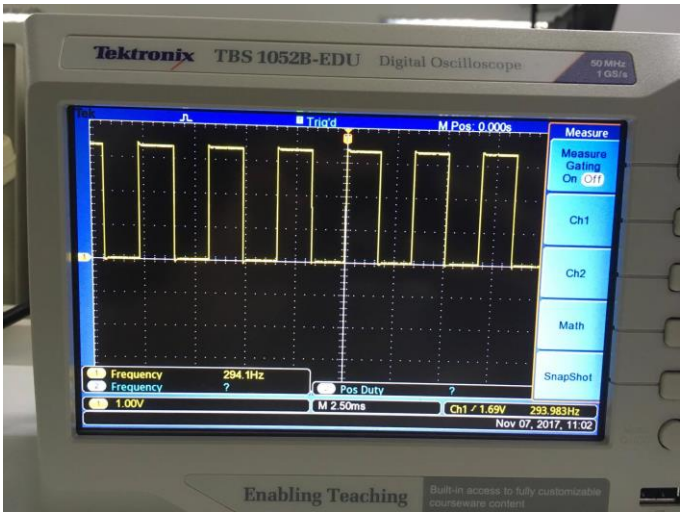
สรุปผลการทดลอง

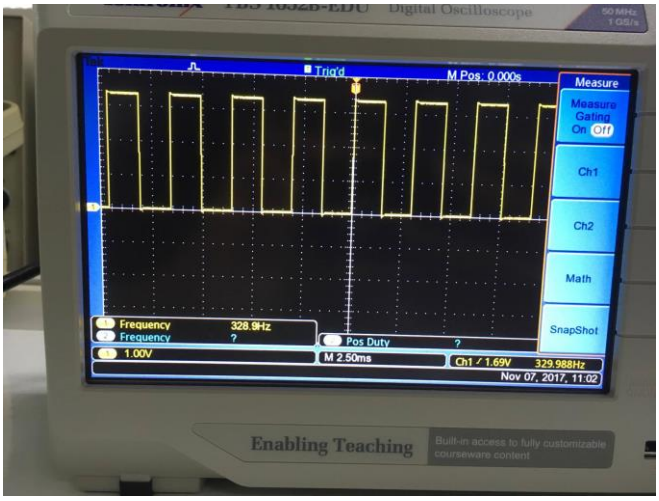
จากการทดลองนั้นเราได้เขียนโค้ดขึ้นมาเพื่อทำให้กดสวิตซ์ทั้ง 8 สวิตซ์เป็นโน๊ตเสียงตั้งแต่ โด เร มี ฟา ซอล ลา ที โดยเราจะกำหนดค่าความถี่ของโน๊ตแต่ละตัวไม่เท่ากันโดยความถี่ของตัวโน๊ตดังภาพประกอบด้านล่าง โดยหลังจากนั้นเราได้ทำการนำตัวแปรตัวโน๊ตที่กำหนดความถี่เรียบร้อยไปใส่ค่าให้กับสวิตซ์ที่เราเซตไว้ดังโค้ดเบื่องต้นดังรูป โดยจะกำหนดดังรูปด้านล่าง และหลังจากนั้นจะมีการวัดคลื่นความถี่ว่าเรานั้นได้ความถี่ตรงกับที่เขียนโค้ดไว้หรือไม่

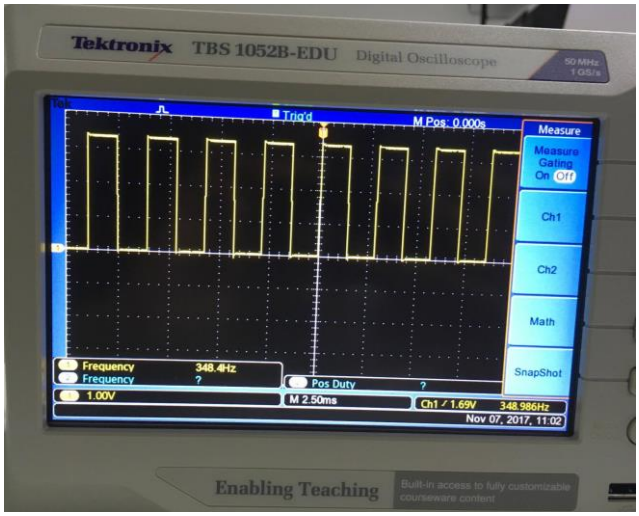
```
parameter doo = (25000000/262/2);
parameter re = (25000000/294/2);
parameter mi = (25000000/330/2);
parameter fa = (25000000/349/2);
parameter sol = (25000000/392/2);
parameter la = (25000000/440/2);
parameter si = (25000000/494/2);
```

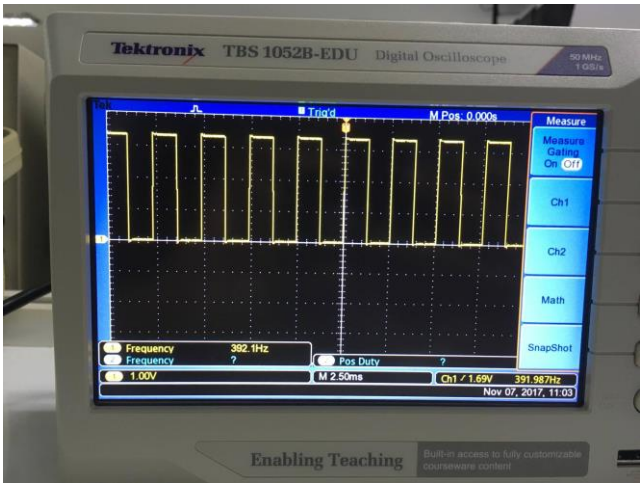
```
case(note)
   7'b0000001 : sp <= s_do;
   7'b0000010 : sp <= s_re;
   7'b0000100 : sp <= s_mi;
   7'b0001000 : sp <= s_fa;
   7'b0010000 : sp <= s_sol;
   7'b0100000 : sp <= s_la;
   7'b1000000 : sp <= s_si;
```

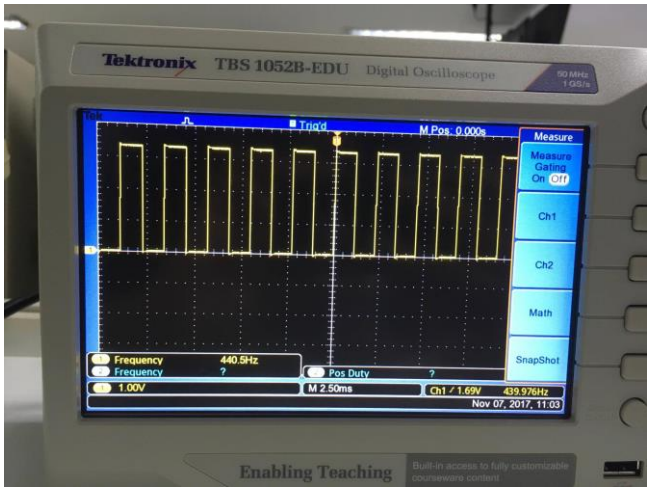

Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 261.7Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 261.990Hz
Nov 07, 2017, 11:02
Enabling Teaching
Built-in access to fully customizable courseware content

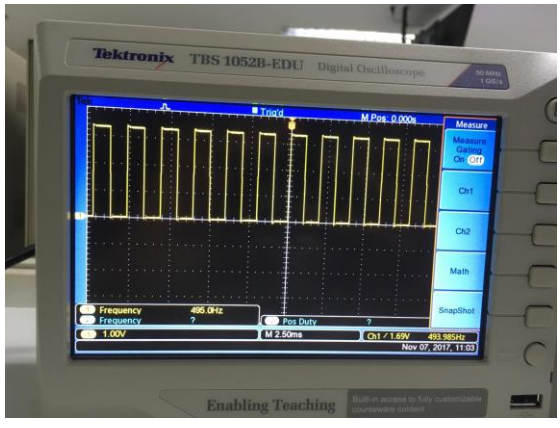

Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 294.1Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 293.983Hz
Nov 07, 2017, 11:02
Enabling Teaching
Built-in access to fully customizable courseware content


Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 328.9Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 329.988Hz
Nov 07, 2017, 11:02
Enabling Teaching
Built-in access to fully customizable courseware content


Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 348.4Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 348.986Hz
Nov 07, 2017, 11:02
Enabling Teaching
Built-in access to fully customizable courseware content


Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 392.1Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 391.987Hz
Nov 07, 2017, 11:03
Enabling Teaching
Built-in access to fully customizable courseware content


Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 440.5Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 439.976Hz
Nov 07, 2017, 11:03
Enabling Teaching
Built-in access to fully customizable courseware content


Tektronix TBS 1052B-EDU Digital Oscilloscope
Trig'd
M Pos: 0.000s
Measure
Measure Gating On Off
Ch1
Ch2
Math
SnapShot
Frequency 495.0Hz
Frequency ?
Pos Duty ?
1.00V
M 2.50ms
Ch1 ∕ 1.69V 493.985Hz
Nov 07, 2017, 11:03
Enabling Teaching
Built-in access to fully customizable courseware content

การทดลองที่ 2

```
21  module c2(
22      input clk,
23      input onoff,
24      input [6:0] note,
25      output speaker
26      );
27
28    parameter doo = (25000000/262/2);
29    parameter re = (25000000/294/2);          //config frequncey
30    parameter mi = (25000000/330/2);
31    parameter fa = (25000000/349/2);
32    parameter sol = (25000000/392/2);
33    parameter la = (25000000/440/2);
34    parameter si = (25000000/494/2);
35
36    reg s_do,s_re,s_mi,s_fa,s_sol,s_la,s_si;
37    reg [31:0] cnt,cnt1,cnt2,cnt3,cnt4,cnt5,cnt6,cnt7;    //config variable
38    reg sp;
39    assign speaker = sp;
40
41    reg [2:0] state = 0;
42    reg sclk;
43    always@(state)                  //case -> in play note
44    begin
45       if(onoff == 0) sp <= 0;
46       else
47       case(state)
48          0 : sp <= s_do;
49          1 : sp <= s_re;
50          2 : sp <= s_mi;
51          3 : sp <= s_fa;
52          4 : sp <= s_sol;
53          5 : sp <= s_la;
54          6 : sp <= s_si;
55          7 : sp <= 0;
56
57       endcase
58    end
```

```
59
60    always @(posedge clk)                    //config clock in each note
61    begin
62       if (cnt1 == 0) begin cnt1 <= doo - 1 ; s_do <= ~s_do; end
63          else cnt1 <= cnt1 - 1;   end
64
65
66    always @(posedge clk)
67    begin
68       if (cnt2 == 0) cnt2 <= re - 1; else cnt2 <= cnt2 - 1; end
69    always @(posedge clk)
70       if(cnt2 == 0) s_re <= ~s_re;
71
72    always @(posedge clk)
73    begin
74       if (cnt3 == 0) begin cnt3 <= mi - 1; s_mi <= ~s_mi; end else cnt3 <= cnt3 - 1; end
75
76
77    always @(posedge clk)
78    begin
79       if (cnt4 == 0) begin cnt4 <= fa - 1; s_fa <= ~s_fa; end else cnt4 <= cnt4 - 1; end
80
81
82    always @(posedge clk)
83    begin
84       if (cnt5 == 0) begin cnt5 <= sol - 1;  s_sol <= ~s_sol; end else cnt5 <= cnt5 - 1; end
85
86
87    always @(posedge clk)
88    begin
89       if (cnt6 == 0) begin cnt6 <= la - 1; s_la <= ~s_la; end else cnt6 <= cnt6 - 1; end
90
91
92    always @(posedge clk)
93    begin
94       if (cnt7 == 0) begin cnt7 <= si - 1; s_si <= ~s_si; end else cnt7 <= cnt7 - 1; end
95
96    always@(posedge clk)                                          //config change state
97    if(onoff == 0||state == 7) state <= 0 ;else
```

```
98    begin
99       if(cnt == 10000000 ) begin state <= state + 1 ; cnt <=0 ; end
100      else cnt <= cnt + 1; end
101
102   endmodule
```

สรุปผลการทดลอง

จากการทดลองนี้จะคล้ายๆในส่วนของการทดลองที่ 1 โดยการทดลองนี้จะเป็นการไล่เสียงตัวโน๊ตโดยเราจะแก้ในส่วนของสวิตซ์ออกไปเป็นส่วนของ state แทนเพื่อให้ทางบอร์ดได้นับ state แทนโดยจะเพิ่มการไล่ตัวโน๊ตไปโดยใช้สัญญาณนาฬิกาเข้ามาช่วยจากโค้ดนั่นเองโดยจะไล่ความถี่ไปเรื่อยๆจนครบแล้วกลับมาเริ่มใหม่

การทดลองที่ 3

```
21  module music(
22        input clk,              //io
23        input onoff,
24        input [6:0] note,
25        output speaker
26    );
27
28    parameter doo = (25000000/262/2);
29    parameter re = (25000000/294/2);          //config frequncey
30    parameter mi = (25000000/330/2);
31    parameter fa = (25000000/349/2);
32    parameter sol = (25000000/392/2);
33    parameter la = (25000000/440/2);
34    parameter si = (25000000/494/2);
35
36    reg s_do,s_re,s_mi,s_fa,s_sol,s_la,s_si;
37    reg [31:0] cnt,cnt1,cnt2,cnt3,cnt4,cnt5,cnt6,cnt7;    //config variable
38    reg sp;
39    assign speaker = sp;
40
41    reg [5:0] state = 0;
42    always@(state)                    //case -> in play note
43    begin
44       if(onoff == 0) sp <= 0;
45       else
46       case(state)
47       0 : sp <= 0;
48       1 : sp <= s_sol;
49       2 : sp <= s_sol;
50       3 : sp <= s_sol;
51       4 : sp <= s_sol;
52       5 : sp <= s_mi;
53       6 : sp <= s_re;
54       7 : sp <= s_mi;
55       8 : sp <= s_sol;
56       9 : sp <= s_do;
57       10 : sp <= s_sol;
58       11 : sp <= s_mi;
59       12 : sp <= s_re;
```

```
60       13 : sp <= s_mi;
61       14 : sp <= s_do;
62       15 : sp <= s_re;
63       16 : sp <= s_do;
64       17 : sp <= s_la;
65       18 : sp <= s_do;
66       19 : sp <= s_do;
67       20 : sp <= s_la;
68       21 : sp <= s_sol;
69       22 : sp <= s_do;
70       23 : sp <= s_do;
71       24 : sp <= s_la;
72       25 : sp <= s_do;
73       26 : sp <= s_do;
74       27 : sp <= s_do;
75       28 : sp <= s_la;
76       29 : sp <= s_sol;
77       30 : sp <= s_do;
78       31 : sp <= s_sol;
79       32 : sp <= s_la;
80       33 : sp <= s_sol;
81       34 : sp <= s_mi;
82       35 : sp <= s_re;
83       36 : sp <= s_do;
84
85       endcase
86    end
87
88    always @(posedge clk)                    //config clock in each note
89    begin
90       if (cnt1 == 0) cnt1 <= doo - 1;
91          else cnt1 <= cnt1 - 1;   end
92    always @(posedge clk)
93       if(cnt1 == 0) s_do <= ~s_do;
94
95    always @(posedge clk)
96    begin
97       if (cnt2 == 0) cnt2 <= re - 1;
98          else cnt2 <= cnt2 - 1; end
```

```
99    always @(posedge clk)
100      if(cnt2 == 0) s_re <= ~s_re;
101
102   always @(posedge clk)
103   begin
104      if (cnt3 == 0) cnt3 <= mi - 1;
105         else cnt3 <= cnt3 - 1; end
106   always @(posedge clk)
107      if(cnt3 == 0) s_mi <= ~s_mi;
108
109   always @(posedge clk)
110   begin
111      if (cnt4 == 0) cnt4 <= fa - 1;
112         else cnt4 <= cnt4 - 1; end
113   always @(posedge clk)
114      if(cnt4 == 0) s_fa <= ~s_fa;
115
116   always @(posedge clk)
117   begin
118      if (cnt5 == 0) cnt5 <= sol - 1;
119         else cnt5 <= cnt5 - 1; end
120   always @(posedge clk)
121      if(cnt5 == 0) s_sol <= ~s_sol;
122
123   always @(posedge clk)
124   begin
125      if (cnt6 == 0) cnt6 <= la - 1;
126         else cnt6 <= cnt6 - 1; end
127   always @(posedge clk)
128      if(cnt6 == 0) s_la <= ~s_la;
129
130   always @(posedge clk)
131   begin
132      if (cnt7 == 0) cnt7 <= si - 1;
133         else cnt7 <= cnt7 - 1; end
134   always @(posedge clk)
135      if(cnt7 == 0) s_si <= ~s_si;
136
137
```

```
138   always@(posedge clk)                                    //config change state
139   if(onoff == 0||state == 37) state <= 0 ;else
140   begin
141      if(cnt == 10000000 ) begin state <= state + 1 ; cnt <=0 ; end
142      else cnt <= cnt + 1; end
143
144 endmodule
```

สรุปผลการทดลอง

จากการทดลองนี้จะเป็นการไล่เสียงตัวโน๊ตให้เป็นเพลง ซึ่งได้ทำการแปลงโค้ดมาจากข้อที่สองโดยการแปลงโค้ดนั้นโดยเราจะเพิ่ม state มากขึ้นโดยขึ้นอยู่กับตัวโน๊ตทั้งหมดในเพลงอย่างเช่นในเพลงช้างของการทดลองที่ 3 นั้นจะมีทั้งหมด 37 state หรือ 37 โน๊ตในเพลงนั่นเองเราเลยเรียง state ดังภาพโค้ดด้านบน